AD

AD-E403 846

Technical Report ARWSE-CR-16003

TACTICAL APPLICATIONS (TACAPPS) JAVASCRIPT FRAMEWORK INVESTIGATION

Craig Klementowski
Tiffany Reid
Norbert Antunes
Jaiden Choong
Liqiao Huang
Ross Arnold

January 2017

U.S. ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER

Weapons and Software Engineering Center

Picatinny Arsenal, New Jersey

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

The citation in this report of the names of commercial firms or commercially available products or services does not constitute official endorsement by or approval of the U.S. Government.

Destroy by any means possible to prevent disclosure of contents or reconstruction of the document. Do not return to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (*From - To*)
January 2017	Final	

4. TITLE AND SUBTITLE

TACTICAL APPLICATIONS (TACAPPS) JAVASCRIPT FRAMEWORK INVESTIGATION

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

Craig Klementowski, Tiffany Reid, Norbert Antunes, Jaiden Choong, Liqiao Huang, and Ross Arnold

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC, WSEC
Fire Control Systems and Technology Directorate
(RDAR-WSF-M)
Picatinny Arsenal, NJ 07806-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC, ESIC
Knowledge & Process Management (RDAR-EIK)
Picatinny Arsenal, NJ 07806-5000

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Technical Report ARWSE-CR-16003

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Tactical Applications (TacApps) Development Team investigated several JavaScript frameworks to facilitate user interface (UI) development in the TacApps Program. Criteria for consideration included: the frequency of change in the JavaScript frameworks' landscape, quality/ease of interaction with other libraries and frameworks, compatibility with Mission Command Data Service (MCDS) design paradigms, productivity, and overall compatibility with TacApps design philosophy. The frameworks explored were Angular JavaScript (AngularJS), jQuery UI, Meteor, Ember, React JavaScript (ReactJS) and Web Components. The team evaluated the benefits, issues and risks of each framework through literature search, comparative study, and actual prototyping. Ultimately, the team decided to move forward with ReactJS as the framework of choice. This is the framework used in the TacApps critical design review prototype demonstration. It is a "view" only framework that eliminates the excess capabilities of other frameworks based on the Model-View-Controller software architectural pattern. The ReactJS also interoperates well with MCDS and other frameworks and JavaScript libraries to provide a robust UI experience for the TacApps user.

15. SUBJECT TERMS
Mission Command Software Tactical Applications (TacApps) Command post client
Command Post Computing Environment (CP CE) Tactical computing environment (TCE)
Mounted computing environment (MCE)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ross D. Arnold
U	U	U	SAR	25	19b. TELEPHONE NUMBER (Include area code) (973) 724-8618

CONTENTS

FIGURES

ACKNOWLEDGMENTS

The authors would like to thank Timothy Rybarski and Gregory Roehrich (U.S. Army Armament Research, Development and Engineering Center, Picatinny Arsenal, NJ) for their sponsorship, and the Tactical Mission Command Product Management Office for funding the Weapons and Software Engineering Center to undertake this effort.

INTRODUCTION

The future of tactical computing demands is an innovative new solution to address a variety of challenging operational needs. As the U.S. Army moves toward a lightweight, fully networked battalion, and disconnected operations, thin client architecture and mobile computing become increasingly essential. Research and development of a hybrid blend of technologies is in progress to address these issues. Together, this blend of technologies and their culmination as a cohesive software product is called Tactical Applications (TacApps).

The TacApps will consist of a web-based Hypertext Markup Language (HTML) 5/cascading style sheets (CSS)/JavaScript user interface (UI) that communicates through a series of application programming interfaces (APIs) to a back-end infrastructure called the Mission Command Data Service (MCDS). Together, these components facilitate a rich collaborative environment. The design of the UI and its methods of communication to the MCDS are critical to the holistic success of the TacApps Program. As TacApps UI development begins, the selection of an appropriate JavaScript framework, or set of frameworks, is likely to impact program development significantly. To mitigate this risk, and select an appropriate framework, members of the TacApps Development Team were tasked to investigate some of the most highly regarded JavaScript frameworks. Several frameworks exist that appear to match TacApps needs. This report explores those frameworks, summarizing the advantages and disadvantages of each. Ultimately, React JavaScript (ReactJS) was determined to be the recommended framework for use in TacApps.

ANGULAR JAVASCRIPT (ANGULARJS)

The AngularJS (commonly referred to as “Angular") is an open-source web application framework maintained by Google, and by a community of individual developers and corporations, to address many of the challenges encountered in developing single-page applications. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model view controller (MVC) architecture, along with components commonly used in rich Internet applications. The Angular provides two-way data binding, meaning developers will not have to write code to synchronize the view to model and vice versa. It is also designed with testing in mind by providing dependency injection (DI) for XMLHttpRequest (XHR), which provides abstraction by allowing developers to sort the model without manipulating the document object model (DOM).

The AngularJS was created as a side project in 2009 by two developers: Misko Hevery and Adam Abrons. It was hosted at GetAngular.com and was originally proprietary software until Hevery and Abrons decided to release it to the open source community due to minimal subscriptions. The original scope of the project was an end-to-end tool that allowed web designers to interact with both the front-end and back-end. After the initial development of AngularJS, Hevery worked on Google Feedback along with two other developers over the course of 6 mos. This amounted to 17,000 lines of code that proved difficult to test and later modify. He proved to his manager that he was able to cut down the lines of code from 17,000 to 1,500 lines within a 3 wk time span. Due to the success of Google Feedback, because of AngularJS, the right parties were put into place to propel the advancement of AngularJS.

Features

- **Two-way Data Binding.** Two-way data binding allows both the model and the view to update each other automatically when either changes. In this manner, the two are always kept synchronized with minimal effort.

- **Directive.** Angular allows for expansion of the HTML syntax by building a directive. A directive is a reusable component that can abstract away complex behavior, DOM, and styling. The Angular comes with many built-in directives such as: ngRepeat, ngBind, ngShow, etc.

- **Expression.** Angular expressions are JavaScript-like code snippets that are enclosed in double curly braces, for example, {{expression}}. Angular will evaluate the expression and output the result. Expressions may contain literals, operators, and variables.

- **Forms.** Controls (input, select, text area, etc.) are ways for users to enter data. A form is a collection of controls for the purpose of grouping related controls together. Form and controls provide validation services so that users can be notified of invalid input before submitting forms. This provides a better user experience than server-side validation alone because users receive instant feedback describing how to correct each error.

- **Module.** Modules allow a developer to easily separate concerns in his/her application. Modules can specify dependencies during their initialization. Modules make unit testing convenient by isolating code into a single testable unit.

- **Routing.** Angular routing allows a developer to map a specific uniform resource locator (URL) to a specific controller. This enables easy bookmarking of URLs which provide specific content. These URLs can also be easily sent to others. Some third-party routing libraries (namely UI-router) allow routing by application state in addition to URL.

- **Scope.** A scope provides the way for controllers and views to interoperate. Both can reference the scope object but not each other, providing isolation, which allows for easier testing. A scope provides a place for expressions to execute, watchers to observe the model, and propagation of events. Scopes are nested in a similar manner to the DOM.

- **Dependency Injection.** Angular implements DI to create components, resolve their dependencies, and provide these dependencies upon request.

- **Filters.** A filter may be used to format data for the view. Filters may be chained together to provide complex, targeted output from input data.

Benefits

- Enables quick prototyping and delivery of dashboard-style (heavy on data-driven forms, charts, lists, and tables) dynamic single-page applications.

- Development may be faster than Knockout, Backbone, and ReactJS frameworks.

- Very expressive and requires less code for the same result as with other libraries.

- Easily testable.

- Good for applications where there is a large amount of highly interactive client side code.

- Two-way data binding.

- DI system.

- Allows effective, componentization of code.

Issues and Risks

- Basics are easy to learn, but then the learning curve becomes very steep.
- Some of AngularJS's best features suffer from a higher level of complexity (DI, services, factories, values, etc.).
- Scopes are easy to use, but not easy to debug.
- Documentation is not up to standard.
- Directives are powerful, but difficult to debug.
- There is a lack of runtime configuration. Most configuration must be done prior to application bootstrap.
- Routing functionality is limited.
- Version 2.0 is not backwards compatible with version 1.x.

JQUERY USER INTERFACE

The jQuery UI is a collective set of UI interactions, widgets, effects, and themes that are built on top of jQuery JavaScript library. The API is simple and intuitive. The jQuery UI elements are found via a query selector with follow-up calls executed upon a resultant set. All widgets have been designed to accommodate the visually impaired, which aligns with the guidance for U.S. Army web applications under Section 508. Widgets that are part of the jQuery UI library are able to function outside of JavaScript environments. Core widgets that are part of the jQuery UI library provide the following interactions in JavaScript and non-JavaScript environments: draggable, droppable, resizable, selectable, and sortable. These behaviors are typical and expected from most modern web applications. The jQuery UI may be used in conjunction with other JavaScript libraries, but as with any JavaScript libraries, a developer may encounter issues when mixing and matching UI components instead of providing all functionality from a single source library.

Features

- **Mobile Ready.** The jQuery UI widgets not only work seamlessly on desktop browsers but mobile browsers as well. The jQuery Mobile is another JavaScript library that is also based on jQuery. The difference in approach is that jQuery Mobile is designed for the mobile platform first and then everywhere else. The reverse is true with jQuery UI (desktop first and then everywhere else). It is also important to note that due to the shared base (jQuery), jQuery Mobile also borrows many widgets from jQuery UI. It is recommended to use either one library or the other but not both; jQueryUI library and jQuery Mobile library should not be co-mingled.
- **Customizable.** The jQuery UI library is customizable prior to download from the jQuery UI website. A developer is able to package only the plugins for the widgets that his/her application requires, resulting in a smaller library footprint. Front-end developers are able to customize UI color scheme by selecting from predefined themes or by providing a

custom theme. Each plugin is customizable via an "options" attribute. A plugin without options delivers default behavior. This functionality is synonymous with specifying CSS styles.

- **Backed by jQuery.** Since jQuery UI is built on top of jQuery, DOM manipulation, Asynchronous Javascript and XML (AJAX) interaction, animation, and event handling are handled with ease. However, this requires basic knowledge in JavaScript, jQuery UI, and jQuery itself in order to facilitate proper, maintainable use of the library.

- **Support for Stateful/Stateless.** Stateful and stateless plugins can be built by using a provided consistent API that is responsible for creating, destroying, setting options, and event support for widgets.

Benefits

- Due to the nature of jQuery UI, and the fact that it inherits its behaviour from jQuery, one of the inherited values is its ease of use when compared to other JavaScript libraries.

- Less lines of code to accomplish a single functionality that might take other libraries several lines to achieve generally resulting in cleaner, more readable code.

- Backed by open source community; many plugins for widgets are available.

Issues and Risks

- HTML elements customized with raw JavaScript might be a better option, depending on the level of customization required by UI items.

- Evolution of the widget set is relatively slow compared to other UI libraries based on jQuery.

- Not a feature rich widget set for complex projects.

METEOR

Meteor was founded in 2011 as a start-up incubated by YCombinator, Mountain View, CA. Meteor is an open-source real-time JavaScript framework written on top of Node.js. The goal is to allow developers to "build apps that are a delight to use, faster than you ever thought possible" (Meteor). It provides everything necessary to build an application rapidly using just JavaScript. In Meteor, UI components are automatically updated as data and stored in the bundled database. Meteor can be used with other JavaScript libraries.

Features

- **One Language (Isomorphic JavaScript).** Both the client and server components of a Meteor application are written in JavaScript. Meteor bundles all the JavaScript files in a project folder and sends them to the server and client. There are times when this behaviour is not desirable, and in those instances, Meteor provides a mechanism to separate the client – server behaviour by use of special directories (e.g., "server" and "client") or by using the Meteor.isClient and Meteor.isServer functions.

    - **Mobile support.** Meteor leverages Adobe PhoneGap and Apache Cordova for mobile device support. This provides the ability to develop cross-platform mobile applications using HTML, JavaScript, and CSS. It also provides the ability to access native device functions, such as a Global Positioning System and camera from your JavaScript code.

- **Database Everywhere.** Meteor provides "full-stack" database drivers to allow the developer to use the same database API on the server and client to access data. Meteor provides an in-memory JavaScript database on the client side. The database APIs are reactive, providing the ability for live database queries. When a query is submitted, the results of the query are returned along with a stream of updates as the results of the query change. Currently, Meteor comes bundled with MongoDB. However, support for other databases are planned for later releases.

- **Data on the Wire.** Meteor uses the distributed data protocol (DDP) for the communication between the client and server. The DDP is a WebSocket-based data protocol that provides the support for fetching structured data from a server and receiving live updates when there are changes made to that data. Rather than sending HTML over the network to the client, the server sends the data and the client renders it.

- **Latency Compensation.** When a client makes changes to the database, Meteor attempts to predict the result of the call on the client side before it receives the response back from the server. This provides the ability to update the UI immediately. Afterwards, when the server response is returned, Meteor will check to see if the prediction was correct. If it was not, it will correct itself and the UI will be updated to reflect the correct response.

- **Full Stack Reactivity.** Meteor Blaze is a library that provides developers the ability to create reactive UIs by writing HTML templates. Inside the templates, developers use template directives, such as "#if" to render portions of the template if a certain condition is met or "#each" to loop through an array. Template tags are also used to access data properties or helper functions. As an example, if a database contained a list of players' names and statistics, those statistics could be referenced in the HTML template using {{name}} and {{statistic}}. By using these two tags, each directive and a helper function, a fully reactive list is displayed with a minimal amount of code that would automatically update as additions, changes, and deletions are made to the list of players.

- **Accounts.** Meteor provides a complete user account system that developers may use in their applications. A developer can provide an out of the box login/registration UI and authentication or customize the UI with very little code. Developers may also use certain packages that Meteor provides to login with common third-party authentication services, such as Google and Facebook.

- **Testing – Velocity.** Velocity is the official testing framework for Meteor. It allows a developer to define tests using several testing libraries, such as Mocha, Jasmine, and Cucumber. The tests are launched automatically as the code is saved and the results are displayed on an HTML overlay once the application is launched.

- **Isobuild.** Meteor Isobuild takes an application's project files (including all dependencies) and processes all the input to generate runnable programs for each of the target platforms (e.g., iOS and Android).

Benefits

- Rapid development of fully reactive applications.
- Built-in account management.
- JavaScript, HTML, and CSS are used for the full application, mobile, client, and server.
- Ability to rapidly create mobile applications from a single code baseline (iOS and Android).
- Meteor provides a package repository for add-on/third-party packages and a command line mechanism for managing the packages for projects.
- Interoperability with other JavaScript frameworks such as React.
- Testing framework supports using existing testing libraries.

Issues and Risks

- Integrating with existing systems or databases that are unable to migrate to the Meteor DB (currently MongoDB) can be difficult and often feels like a hack. As expected, the latency compensation benefits that Meteor provides are lost in this case. This poses a very serious issue for TacApps, as MCDS does not use MongoDB, and therefore does not integrate well with Meteor.
- Server-side rendering is not supported out of the box.

EMBER

The Ember.js framework was created by Yehuda Katz, a member of the jQuery, Ruby on Rails, and SproutCore core teams to advocate the ideology of convention over configuration. Ember.js was originally a fork of SproutCore known as SproutCore 2.0. It eliminated the UI widget aspect of SproutCore and instead focused on the MVC portion of the SproutCore framework. The framework's name changed to Amber.js, but due to a naming conflict with another framework (focused on Smalltalk), it was later renamed to Ember.

The Ember.js is an open-source, client-side JavaScript web application framework based on the MVC software architectural pattern (fig. 1). It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, router for managing application state, and automatically updated templates powered by its own template library known as HTMLBars. HTMLBars is built on top of Handlebars.js, a template processor for dynamically generating HTML pages.

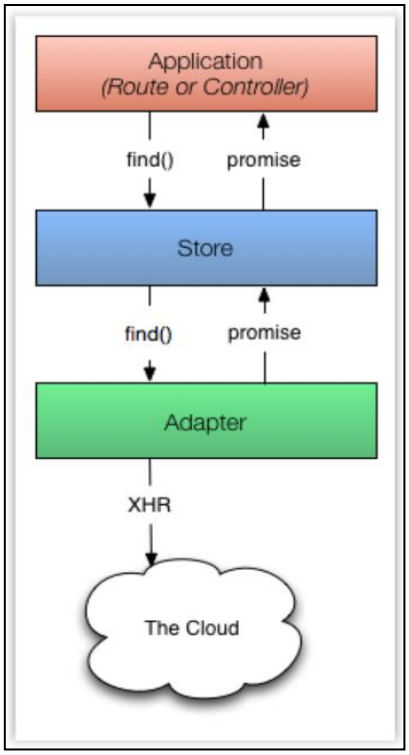

Figure 1
Ember architecture

Features

- **Ember Data.** Most Ember.js applications use Ember Data, a data persistence library that maps client-side models to server-side data. Ember Data, by default, has the ability to load and persist records and their relationships without any prior configuration via a representational state transfer (RESTful) JavaScript object notation (JSON) API. Ember Data integrates tightly with Ember.js to easily retrieve records from a server, cache them for performance, persist updates to the server, and create new records on the client. This API provides many of the facilities with an object-relational mapping, but it is also easily configurable to any server with the use of adapters. It is possible to use Ember.js without Ember Data.

- **Ember Inspector.** The Ember Inspector (fig. 2) is an extension currently available for Mozilla Firefox, Internet Explorer, and Google Chrome web browsers that further assists in debugging Ember applications.

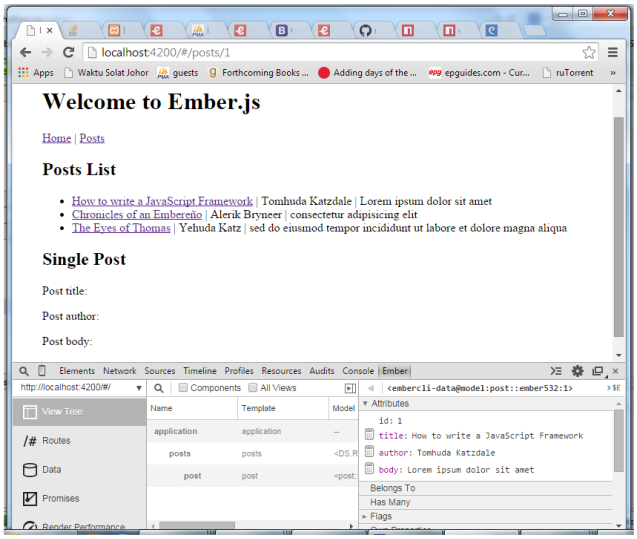

Figure 2
Ember Inspector

Features of the inspector include, but are not limited to:

- View all of the routes defined in an application.
- Reference Ember's naming conventions for URLs, including what to name controllers, templates, routes and more.
- Overlay an application with information about which templates, controllers, and models are currently being rendered.
- Inspect the objects in an application, such as models and controllers, with UI that fully supports Ember features such as bindings and computed properties.
- Access to an application's objects from the console is readily available via the use of the "$E" variable.
- If using Ember Data, the capability to see all of the records that have been loaded.

- **Ember Command Line Interface (CLI).** The Ember CLI utility is a build tool for Ember.js based on Broccoli. Broccoli is a back-end-agnostic build tool focused on "rebuilding": identifying and watching exact files that require a rebuild. Features of Ember CLI include, but are not limited to:
  - ECMAScript 6 module transpiler, converts ECMAScript 6 code into "RequireJS-style" modules.
  - QUnit, Ember Testing and Ember QUnit ready applications which facilitates authoring of unit and integration tests.

- Dependency management, uses the Bower package manager making it easy to keep dependencies up to date.
- Runtime configuration using a JSON configuration file named .ember-cli in user home directory to facilitate the inclusion of command line options.
- Content security policy guards application against the risks of cross-site scripting attacks.
- Supports node or io.js and node package manager.
- Generators based on blueprints.
- Ability for third-party extensions via add-ons.

Benefits

- Ember offers structure necessary for large and small projects.
- Build automation with Ember-CLI.
- Two-way UI data binding via a declarative approach to automatically update the view layer when the underlying model changes.
- Composite views pattern allows modular, reusable code and rich view component hierarchy.
- Web presentation layer based on HTML and CSS.
- Interoperability with other JavaScript frameworks such as jQuery.

Issues and Risks

- Ember data is currently in Beta and its API should not be considered stable until version 1.0.

REACT JAVASCRIPT

Facebook created React internally and released it as an open source project. Facebook is using React in most of its current projects (e.g., Instagram.com). Khan Academy, among other popular websites, also uses React. React provides the view (V) in a MVC architecture. It can be used with other JavaScript frameworks and does not dictate the overall structure of a web application. This is considered advantageous for TacApps, as the code will be more resilient against framework obsolesce if the framework does not dictate its structure.

Java Serialization to XML

In order to benefit fully from React, code and HTML are written in a new language called Java Serialization to XML (JSX). The JSX is a JavaScript extension in which a developer can embed HTML and CSS within the JavaScript code. This allows for rapid development of a single feature of a single component in a single file. The JSX code is “compiled” into normal JavaScript to enhance

performance while running in the browser. The output code will also run in legacy browsers such as Internet Explorer 8 (with some minor issues).

Flux

Facebook has also introduced the Flux application architecture along with React. There is a Facebook-provided Flux library as well as other open source implementations of the Flux pattern. Flux introduces a unidirectional data flow with changes propagating via Action, Dispatcher, Store, and View. The flow in one direction simplifies the application. A store contains the data and application logic for a logical domain. Stores register with a dispatcher. A Dispatcher will then call a Store call-back function upon receiving updates. The Store then emits a change event to the V. The Store has no setters and only operates on getters (fig. 3). Facebook believes that using React with Flux has rapidly simplified and sped up their development process.

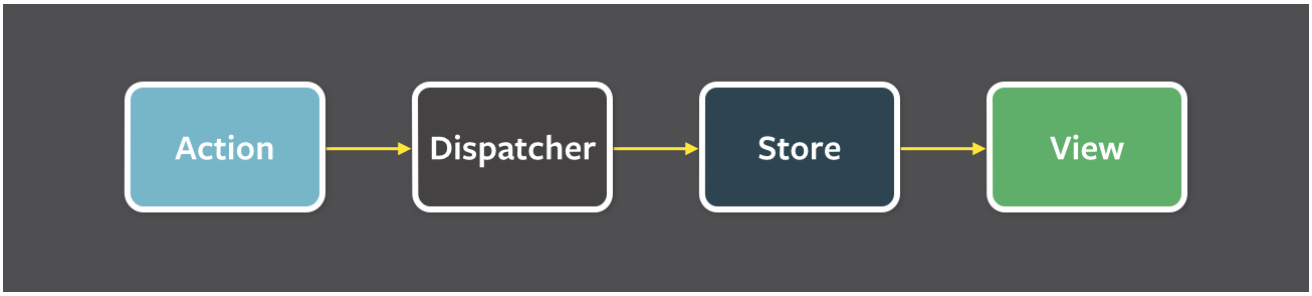

Figure 3
Basic Flux flow

React Native

Facebook is creating additional libraries that will allow compilation of React code into Native applications. Their guiding principle is “Learn once, write anywhere.” Note, this is different from the “Write once, run anywhere” principle that Java is striving for. Once React is mastered, applications can be easily written without having to learn specific device native code. Native applications will by definition be more performant and offer a better user experience than pure web applications. The intentional separation of concerns between virtual DOM and rendering allows the rendering component to be replaced.

As of May 1, 2015, React Native for iOS has been released. React Native for Android is said to be 6 mos. away. It would seem that React Native for Android would allow development of Android applications in a similar fashion to that of a web application.

React was used in the TacApps critical design review demonstration. The MCDS was used to provide the data access layer (model). React components tied the template HTML and CSS (view) with the application logic (controller) together into a neat bundle. The pairing of MCDS with React seemed very natural and complementary.

Benefits

- React uses a virtual DOM that allows it to be smart about updating the actual DOM. React will only update what is required. This form of intelligent rendering leads to exceptional performance. It also frees the developer from tedious and bug prone DOM updates.

- React documentation and videos talk about empowering declarative programming and avoiding imperative programming. Developers should not be focusing on the minutiae about what exact DOM elements need to be modified but rather describing what they want to happen and letting React handle the grunt work.

- React is built from the ground up to allow (but not require) server-side rendering and to support search indexing.

- Since React is a "view" framework, React can be used with almost all other JavaScript frameworks.

- React gives the developer appropriate error messages that describe what the problem is, what line it is on, and, in many instances, exactly how to fix the issue.

- React Native project is underway to allow React code to be compiled into native applications.

- Using JSX enables single file componentization of a view element.

- React provides a way to use third-party libraries like jQuery plugins.

- React Developer Tools add-on is available to facilitate debugging in the Chrome browser.

Issues and Risks

Framework lock-in is high with React, although this is also the case with other frameworks. If support for React ceases or the framework becomes obsolete, TacApps UI rework will be required. However, using a complete MVC framework like AngularJS or Meteor would likely require a much larger rewrite.

WEB COMPONENTS

Web Components are defined by a set of proposed web standards that enable developers to create reusable widgets. There are four specifications that are part of Web Components: Custom Elements, HTML imports, HTML templates, and Shadow DOM. The HTML templates currently exist as World Wide Web Consortium (W3C) group notes. Custom Elements, shadow DOM, and HTML imports are currently W3C working drafts. Although the standards for Web Components are incomplete, Google Chrome browser still offers full support. Google also offers the Polymer library that is built on top of Web Components and offers integrated Polyfills. Polyfills are JavaScript libraries that implement features a particular browser does not support.

The HTML templates allow for the declaration of inert DOM subtrees in HTML and the manipulation of them to instantiate document fragments. The HTML templates essentially can be used and reused to create identical content throughout the application.

Custom Elements allow an author to define and use new types of DOM elements in a document. By using Custom Elements, HTML element extension is possible along with component reuse. Custom Elements also support lifecycle callbacks on: attribute change, creation, attachment, and detachment.

Shadow DOM allows a developer to separate content and presentation. Used in conjunction with HTML templates, it allows the attachment of a “phantom” like DOM in an HTML element where styling of the element is specific to the Shadow DOM and not the entire document. The reverse is true where styles that are applied to the Shadow DOM are not automatically applied to the rest of the document. This proves useful in the event of element ID and/or style class name reuse for templating purposes. It is important to note that the rendering of documents that include Shadow DOMs will render the original document DOM only to the point of Shadow DOM(s) insertion at which the subtree of the original DOM will be replaced with the subtree of the Shadow DOM (fig. 4).

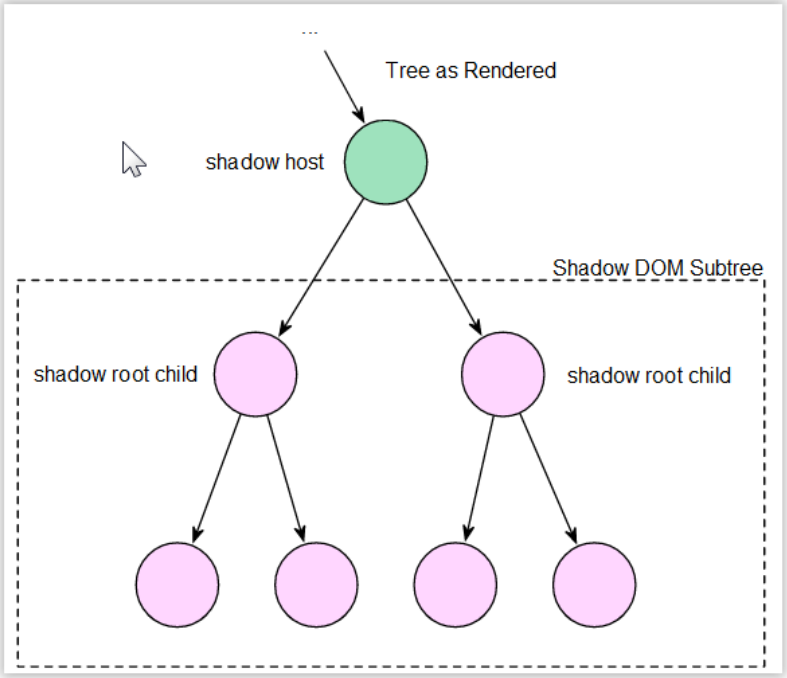

Figure 4
Shadow DOM tree hierarchy

The HTML Imports lends reuse of script, document (HTML), and style (bundled as a single resource) in other HTML documents. An HTML file is parsed synchronously. Using HTML Imports via the “link” element, deferment of immediate script execution occurs while order of script execution is maintained. For example, if an import of HTML page 2 occurs at line 2 of HTML page 1, and within HTML page 2 it includes the import via “script” tags of two different JavaScript files, those files are imported in linear order. Inside HTML page 1, the parsing of line 3 is **not** withheld until the line 2 has completed. There is also an option to prevent the parser blockage by using the “async” attribute on the “link” element. The result is immediate parsing of the proceeding line. Another benefit of using HTML Imports is to limit network connections (calls) for identical resources. If HTML page 1 contains an import via the “link” element to **file1.js** and HTML page 2 contains the same link, **file1.js** is not fetched twice. The only caveat is that the location of **file1.js** must be within the same domain or origin.

Many people believe that Web Components may in fact become the future of the web, assuming browser support becomes universal (fig. 5). While it is possible that this may come to pass, Web Components are not yet mature enough to recommend for use in TacApps.

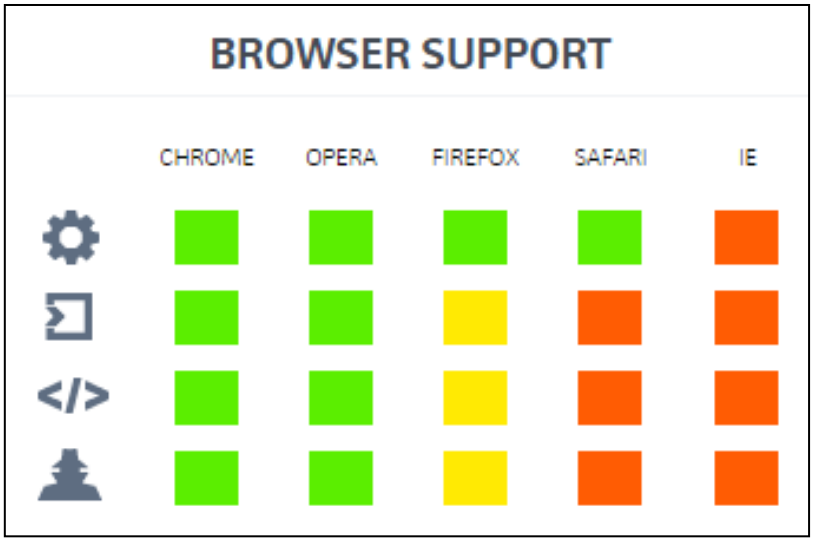

Figure 5
Web Components browser support

Benefits

- Not surprisingly, Web Components allow developers to componentize their code.
- Web Components are supported at the native browser level and do not require additional tools (when supported by a particular browser).
- Lack of current browser support can be addressed using a Polyfill (a JavaScript library that implements the missing feature).
- Shadow DOM allows CSS scoping, i.e., style names will not conflict with other libraries that use the same name for a style.

Issues and Risks

- Web Components specification is not yet complete, and Google Chrome is the only browser that offers first class support.
- Internet Explorer offers no native support for Web Components.
- Microsoft Edge browser has not yet committed to any of the four specifications.
- Firefox offers some support, but currently Web Components support must be turned on in browser options.
- Code can be difficult to comprehend; on par with understanding an XML document.
- Web Components are not widely used in production environments.

CONCLUSIONS

Several key factors stood out during the assessment of the various frameworks. The Angular JavaScript (AngularJS), although quite popular for rapid prototyping, does not appear to be a good choice for a production system. The latest version of AngularJS (AngularJS 2.0) has not been officially released (currently under developer preview), is not production ready, and is not backwards compatible with version 1.0. Therefore, if AngularJS was chosen for use in Tactical Applications (TacApps), any code written for the latest stable version (1.3) would need to be rewritten for 2.0 in the future. Also, AngularJS is considered to be difficult to learn relative to other frameworks. For these reasons, it is not an appropriate selection for TacApps.

The other frameworks were also unsuitable for various reasons. The jQuery user interface [UI (jQueryUI)] at first appeared to be a viable option mainly due to its jQuery roots, which bring a large developer community with it. However, the actual evolution of the jQueryUI widget set is slow and it does not provide a feature-rich widget set. Also jQueryUI is not a “mobile-first” framework, thus user interface (UI) work would need to be duplicated to support desktop and mobile versions of TacApps. Some of the other frameworks focus on data synchronization between the UI and the database. TacApps does not need this type of synchronization, as the Mission Command Data Service (MCDS) provides this capability. Ember and Meteor are thus eliminated as viable options. Web Components provides a set of proposed web standards to facilitate the creation of reusable widgets. However, not all web browsers have fully supported web components. The only web browsers that do support this standard are Google Chrome and Opera. This is unacceptable for TacApps as the Army Gold Master at the time of TacApps fielding will likely ship with some version of Internet Explorer, FireFox, and/or Microsoft Edge.

Ultimately, after examining the JavaScript frameworks described throughout this report, the TacApps Development Team has decided to move forward with the React JavaScript (ReactJS) framework. The ReactJS was used in the initial prototype critical design review demonstration. ReactJS is a “view only” framework, meaning it does not provide the “M” nor “C” in the model view controller architectural software pattern as do some of the other frameworks. Dependency upon the MCDS eliminates the need for modelling and controlling components of a JavaScript framework and allows development of the UI to focus on the view. The ReactJS appears to be a suitable fit in this regard, as it fulfils the needs of the TacApps UI without providing an excess of unneeded code and functionality. It also allows seamless interaction with other JavaScript frameworks and libraries. ReactJS would be best paired with a cascading style sheets library or libraries to tailor fit the look and feel of the desired TacApps UI.

BIBLIOGRAPHY

Frameworks

1. Selle, P., Ruffles, T., Hiller, C., and White, J., Choosing a JavaScript Framework, Bleeding Edge Press, available from http://techbus.safaribooksonline.com/book/programming/javascript/9781939902085, 2014.

2. TodoMVC|Helping you select an MV* framework, retrieved May 11, 2015.

AngularJS

3. Austin, A., An Overview of AngularJS for Managers. Retrieved from http://andrewaustin.com/an-overview-of-angularjs-for-managers, August 27, 2014.

4. Chemel, R., Pros and Cons of AngularJS, retrieved from http://blog.backand.com/pros-and-cons-of-angularjs, April 15, 2015.

5. Guide to AngularJS Documentation, (n.d.), retrieved from https://docs.angularjs.org/guide.

6. Herskovits, I., retrieved May 8, 2015, from http://blog.backand.com/5-things-starting-angularjs, September 23, 2014.

7. Howard, J., retrieved from https://www.youtube.com/watch?v=Ja2xDrtylBw, January 14, 2013.

8. Kalveram, M., How using Angular.js can benefit your startup. Retrieved from https://logbook.hanno.co/how-using-angular-js-can-benefit-your-startup/, January 26, 2015.

9. Migutsky, A., Two years with Angular, retrieved from https://medium.com/@mr_mig_by/2-years-with-angular-b72a81f9e1ae, November 24, 2014.

10. Wahlin, D., retrieved from https://www.youtube.com/watch?v=i9MHigUZKEM, April 12, 2013.

Ember.js

11. Ember.js. GitHub, retrieved from https://github.com/emberjs.

12. Ember, web, retrieved from http://emberjs.com/, 2015.

13. Models, Ember, retrieved from http://guides.emberjs.com/v1.10.0/models/, 2015.

14. Overview, Ember CLI, retrieved from http://www.ember-cli.com/.

15. Hempton, Gordon L., The Top 10 Javascript MVC Frameworks Reviewed, retrieved from http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/, January 14, 2014.

16. Khosravy, Pooyan, Ember Data: A Comprehensive Tutorial for the ember-data Library, retrieved from http://www.toptal.com/emberjs/a-thorough-guide-to-ember-data.

17. Shan, Paul, EmberJS Tutorial: Two Way Data Binding, retrieved from http://voidcanvas.com/emberjs-tutorial-two-way-data-binding/, January 28, 2014.

BIBLIOGRAPHY
(continued)

jQuery UI

18. jQuery/jQuery UI, GitHub. (2015) retrieved from https://github.com/jquery/jquery-ui, 2015.

19. jQuery User Interface, retrieved from https://jQuery UI.com/, 2015.

20. Getting Started with jQuery UI, retrieved from https://learn.jquery.com/jquery-ui/getting-started/, April 17, 2015.

21. jQuery Mobile, retrieved from https://jquerymobile.com/, 2015.

Meteor

22. Blaze Meteor Reactive Templating library, (n.d.), retrieved from https://atmospherejs.com/meteor/blaze.

23. Dascalescu, D., (April 30, 2015), Why Meteor, retrieved May 11, 2015, from http://wiki.dandascalescu.com/essays/why_meteor.

24. Meteor, (n.d.), retrieved from www.meteor.com.

25. Principles of Meteor, (n.d.), retrieved from http://docs.meteor.com/#/full/sevenprinciples.

26. Velocity Testing Meteor Applications, (n.d.), retrieved from http://velocity.meteor.com/.

ReactJS

27. A JavaScript Library for Building User Interfaces|React, retrieved from https://facebook.github.io/react/index.html, May 11, 2015.

28. Facebook/Flux, retrieved from https://github.com/facebook/flux, May 11, 2015.

29. Facebook/React, retrieved from https://github.com/facebook/react, May 11, 2015.

30. Facebook/React-native, retrieved from https://github.com/facebook/react-native, May 11, 2015.

31. Flux|Application Architecture for Building User Interfaces, retrieved from https://facebook.github.io/flux, May 11, 2015.

32. Manalang, R., Rebuilding HipChat with React.js., retrieved from https://developer.atlassian.com/blog/2015/02/rebuilding-hipchat-with-react/, February 10, 2015.

33. McKeachie, C., React.js and How Does It Fit In With Everything Else?, retrieved from http://www.funnyant.com/reactjs-what-is-it/, July 23, 2014.

34. Pełka, M., 6 Reasons Why We Love React.JS, retrieved from http://www.syncano.com/reactjs-reasons-why-part-1/, 2015.

BIBLIOGRAPHY
continued

35. Occhino, T., React Native: Bringing modern web techniques to mobile, retrieved from https://code.facebook.com/posts/1014532261909640/react-native-bringing-modern-web-techniques-to-mobile, March 26, 2015.

36. React Native|A framework for building native apps using React, retrieved from https://facebook.github.io/react-native, May 11, 2015.

Web Components

37. Developer Resources: MSEdge Dev., retrieved from http://dev.modern.ie/platform/status/, May 11, 2015.

38. Dodson R., A Guide to Web Components, retrieved from https://css-tricks.com/modular-future-web-components, September 9, 2014.

39. Internet Explorer Web Platform Status and Roadmap - status.modern.IE, retrieved from https://status.modern.ie/, May 11, 2015.

40. Leggetter, P., Why you should be using Web Components now. And how, retrieved from http://webcomponents.org/presentations/why-you-should-be-using-web-components-and-how-at-devweek/, March 24, 2015.

41. Polymer, retrieved from https://www.polymer-project.org, May 11, 2015.

42. Polymer/polymer, retrieved from https://github.com/Polymer/polymer, May 11, 2015.

43. WebComponents.org: a place to discuss and evolve web component best-practices, retrieved from http://webcomponents.org/, May 11, 2015.

44. Web Components|MDN, retrieved from https://developer.mozilla.org/en-US/docs/Web/Web_Components, May 11, 2015.

Further Reading

http://en.wikipedia.org/wiki/Ember.js
https://addons.mozilla.org/en-US/firefox/addon/ember-inspector/
http://en.wikipedia.org/wiki/JQuery_UI

DISTRIBUTION LIST

U.S. Army ARDEC
ATTN: RDAR-EIK
RDAR-WSF-M, C. Klementowski
T. Reid
N. Antunes
J. Choong
L. Huang
R. Arnold
Picatinny Arsenal, NJ 07806-5000

Defense Technical Information Center (DTIC)
ATTN: Accessions Division
8725 John J. Kingman Road, Ste. 0944
Fort Belvoir, VA 22060-6218

GIDEP Operations Center
P.O. Box 8000
Corona, CA 91718-8000
gidep@gidep.org

REVIEW AND APPROVAL OF ARDEC TECHNICAL REPORTS

TacApps JavaScript Framework Investigation

Title — Date received by LCSD

Ross Arnold

Author/Project Engineer — Report number (to be assigned by LCSD)

x8618 | 31 | RDAR-WSF-M

Extension | Building | Author's/Project Engineers Office (Division, Laboratory, Symbol)

PART 1. **Must be signed before the report can be edited.**

a. The draft copy of this report has been reviewed for technical accuracy and is approved for editing.

b. Use Distribution Statement A__✓__, B____, C____, D____, E____, F____ or X____ for the reason checked on the continuation of this form. Reason: _____ public release _____

1. If Statement A is selected, the report will be released to the National Technical Information Service (NTIS) for sale to the general public. Only unclassified reports whose distribution is not limited or controlled in any way are released to NTIS.

2. If Statement B, C, D, E, F, or X is selected, the report will be released to the Defense Technical Information Center (DTIC) which will limit distribution according to the conditions indicated in the statement.

c. The distribution list for this report has been reviewed for accuracy and completeness.

Patti Alameda

Division Chief — 3/7/16 (Date)

PART 2. To be signed either when draft report is submitted or after review of reproduction copy. This report is approved for publication.

Patti Alameda

Division Chief — 3/7/16 (Date)

Andrew Pskowski

RDAR-CIS — 2/1/17 (Date)

LCSD 49 supersedes SMCAR Form 49, 20 Dec 06